Original Investigations JAMIA

Model Formulation ■

Generic Data Modeling for Clinical Repositories

STEPHEN B. JOHNSON, PhD

Abstract **Objective:** To construct a large-scale clinical repository that accurately captures a detailed understanding of the data vital to the process of health care and that provides highly efficient access to patient information for the users of a clinical information system.

Design: Conventional approaches to data modeling encourage the development of a highly specific data schema in order to capture as much information as possible about a given domain. In contrast, current database technology functions most effectively for clinical databases when a generic data schema is used. The technique of "generic data modeling" is presented as a method of reconciling these opposing views of clinical data, using formal operations to transform a detailed schema into a generic one.

Results: A complex schema consisting of hundreds of entities and representing a rich set of constraints about the patient care domain is transformed into a generic schema consisting of roughly two dozen tables. The resulting database design is efficient for patient-oriented queries and is highly flexible in adapting to the changing information needs of a health care institution, particularly changes involving the collection of new data elements.

Conclusion: Conventional approaches to data modeling can be used to develop rich, complex models of clinical data that are useful for understanding and managing the process of patient care. Generic data modeling techniques can successfully transform a detailed design into an efficient generic design that is flexible enough to meet the needs of an operational clinical information system.

■ **JAMIA.** 1996;3:328–339.

Caring for a patient is a complex process that involves many different professionals, organizations, and materials. A key strategy pursued by many health care institutions is to collect as much information as possible about the process as it unfolds in order to maximize sharing among the many participants, deepen understanding of the process, make improvements, and reduce cost. Different institutions have different levels of computerization, and they manage information in different ways,[1–6] so the term "clinical repository" is used in this article to denote a shared resource of patient data for the purpose of clinical care (in both the inpatient and outpatient settings), and the term "clinical information system" is used to refer to the collection of computer applications that collect, process, and display information maintained in the clinical repository.

The construction of a clinical repository has two main objectives:

Affiliation of the author: Department of Medical Informatics, Columbia University, New York, NY.

Correspondence and reprints: Stephen B. Johnson, PhD, Department of Medical Informatics, Columbia University, 161 Fort Washington Avenue, DAP 1310, New York, NY, 10032. e-mail: stephen.johnson@columbia.edu

Received for publication: 1/15/96; accepted for publication: 5/13/96.

- establishing a clear understanding of the data that are relevant to the health care process, and
- implement a database that performs efficiently for patient care tasks.

The first objective is the province of data modeling,[7,8] and the second is the purpose of database design.[9,10] Both of these disciplines are necessary for building a successful clinical information system; each faces different obstacles.

The goal of this paper is to reconcile the objectives of data modeling with the techniques of database design that have been found to be successful in production-oriented clinical information systems. The technique of "generic data modeling" is presented as a means of obtaining an efficient patient database design from the detailed descriptions that result from conventional approaches to data modeling. Formal methods are described that permit the transformation of a specific schema (useful as a high-level model of an institution's data) into a generic schema (useful for implementation as a working database). This transformation helps bridge the gap between data modeling and database design, and it makes explicit how these two disciplines are related in creating operational clinical information systems.

Background

Data modeling produces a formal description of the data of a given enterprise—a "conceptual schema." This formal description represents concepts of interest to that domain (people, places, equipment, events, etc.), and it indicates how these entities are conceptually related to one another (roles played in events, membership in organizations, ownership of equipment, location of physical objects, etc.). To be useful to an enterprise, the conceptual schema must represent only those facts and constraints in a domain that are known to be true and on which there is agreement. Thus, the more detailed the conceptual schema, the more is known about the domain, and the greater the degree of agreement among the participants in the enterprise.

Data modeling in health care[11,12] is a difficult and time-consuming task because of the vastness of the domain (a very large number of distinct concepts and ways in which these concepts may be related to one another), the complexity of the knowledge, and the wide variety of participants, all with slightly differing views about what the process is (physicians, nurses, therapist, technicians, administrators, clerks, etc.). Current understanding of the process of health care is incomplete, and consensus is lacking in certain areas, which limits the degree of formalization that is possible.

Database design seeks to produce a database that meets the needs of an enterprise by using available computer technology to implement the conceptual schema.[7-10] The choice of technology can have an enormous impact on what can be achieved in an information system. In industry today, there is a wide range of technologies in use: indexed files, hierarchical databases, relational databases, object-oriented databases, and heterogeneous systems that may combine several of these technologies.[13] This paper focuses on design issues for relational databases because the relational model is relatively easy to understand and because this technology is currently the most robust and pervasive for large-scale, production-oriented information systems.[13-15]

Database design for patient care systems must address two important requirements: rapid retrieval of data for individual patients, and adaptability to the changing information needs of an institution (new applications, new queries and updates, new data elements, etc.). A patient's data must be retrievable as quickly as possible (less than one second for a typical transaction); the needs of the patient demand immediate intervention, and the responsibilities of health care personnel allow only limited time to interact with an information system. Adaptability is necessary because formalized knowledge of the health care process is always incomplete and evolving. As a result, computer applications undergo frequent enhancement, and new computer systems are constantly being integrated. The clinical information system must provide a method to keep up with this environment of constant change without affecting the performance of the clinical repository. It would be unacceptable, for example, to shut down the system several times a month in order to accommodate application changes.

Given the current state of the art in relational database technology, the objectives of data modeling and database design for patient care are in conflict. Although a highly detailed conceptual schema helps promote consistency and understanding of the data used by an enterprise, such a design will perform poorly when implemented because a highly detailed schema will be implemented as a large number of tables. This means that a given query may have to combine data from a many different tables by performing "join" operations, which greatly increase query execution time.[9,10] Moreover, a highly detailed schema tends to be very sensitive to change because individual col-

Table 1 ■

Patient Table with Specific Columns

Patient ID	Birthdate	Sex	SSN	Address	Telephone	Person ID
12345	06-22-95	M	123-45-6789	123 E 45 St., NY, NY	212-555-1234	112233
22222	02-29-52	F	222-22-2222	22 W 22 St., NY, NY	212-555-2222	987654

umns must be added or re-defined. Changes to the repository schema have been shown to significantly increase the amount of software maintenance required for the applications that access the repository.[14,16] In addition, each alteration to the schema degrades performance, ultimately requiring that the database's physical structure be reorganized on disk.[9,10] In a large hospital, in which millions of entries for visits, drugs, laboratory tests, etc., are made to the repository each year, reorganization of database tables may require significant down time (hours or even days).

An alternative strategy is to employ a generic schema for the clinical repository.[17–20] A generic schema has a small number of generalized concepts and thus will be implemented as a database with only a few tables. In such a design, related data tend to be present in the same table, obviating the need for expensive join operations. In addition, the design is highly flexible, since different values or "codes" are used to indicate different data elements instead of specific column names.[21] Applications can store new values in the tables without having to alter the repository schema. Tables 1 and 2 illustrate this difference in flexibility for patient demographic data. Adding a new demographic element such as "ethnicity" requires adding a new column to Table 1. In contrast, the design of Table 2 requires only that a new code for ethnicity be defined to store in the Demographic-Type column; no change to the table schema is needed.

Using generic patient databases and coded values has a long and important tradition in health care information systems.[1–6] Although many implementations have been chosen, each system employs a generic schema for the patient database and some form of "data dictionary" to manage the coded data elements that can be stored in the patient database.[22,23] This design was crucial to achieve adequate performance for patient-oriented queries and to provide flexibility for change.[24,25]

Table 2 ■

Demographic Table with Generic Columns

Patient ID	Demographic-Type	Demographic-Value
12345	Birthdate	06-22-95
12345	Sex	M
12345	SSN	123-45-6789
12345	Address	123 E 45 St., NY, NY
12345	Telephone	212-555-1234
12345	Person ID	112233

Methods

The basic steps of generic data modeling for a clinical information system are:

1. Develop a detailed schema of the medical data that will be managed by the system.
2. Filter out concepts and relations that do not vary across patient records.
3. Transform the detailed schema into a generic schema.
4. Implement the generic schema using a database management system.

These steps are explained in the sections below. The technique is presented from the perspective of designing a new database. However, methods are also useful in other data engineering tasks, such as integrating legacy database systems and maintaining operational repositories. Since reverse-engineering and system maintenance are complex topics in themselves, they will not be discussed further in this paper.

Conceptual Schema for Clinical Data

A conceptual schema for patient care is a representation of the data required to manage the health care process. The developers of a clinical repository construct this "community view" of the institution's data by integrating the many local views held by physicians, nurses, administrators, researchers, etc. This process of integration means resolving many discrepancies and making compromises to produce the best possible statement about the data on which the institution's members can agree.

This data modeling activity can benefit from using a very high-level model (formalism) with rich seman-

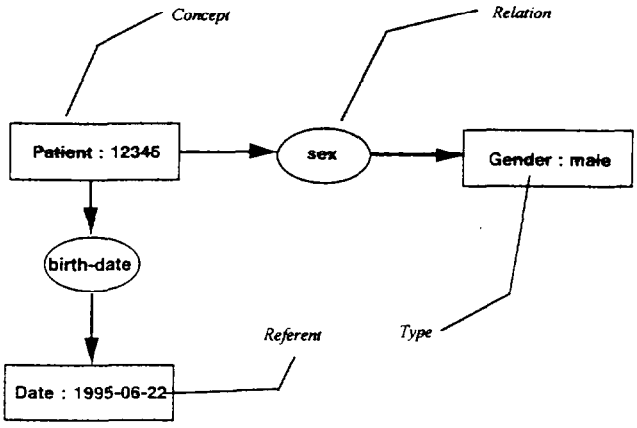

Figure 1 Example of a conceptual graph in the graphical format. The graph depicts a patient whose sex is male and whose birth date is 6/22/1995. A concept may be connected to other concepts by one or more relations. Each concept has a type, which denotes the category to which it belongs, and a referent, which identifies a specific individual within that category.

tics. In the data modeling phase, developers should be concerned with data representation, and not with issues of data storage and access efficiency. The conceptual schema should correspond as closely as possible to expert understanding of the medical domain. Concepts in the schema must use terms that are readily understandable by medical personnel, and the schema should be free of computer jargon. One way to confirm these properties is to ensure that the model can be easily translated into natural language sentences that can be verified by domain experts.

Such a schema typically has many concepts and a rich classification structure that provides many specializations and generalizations of medical concepts. A given concept may belong to many different classes. The semantics of the model are carried by a rich set of connections between concepts (e.g., part–whole structures) that define how concepts in the medical domain are related to one another.

This paper presents a simple conceptual schema for clinical data. This schema is only a small fragment of what would be required in a real clinical information system. The formalism of Conceptual Graphs (CG)[26] was chosen for several reasons. A general-purpose graphic notation (Fig. 1), CG can represent everything that can be modeled using the Entity Relationship[27] or Structural Model,[28] both useful notations for database modeling. In addition, CG has a standard linear notation that uses conventional typewriter symbols (Fig. 2) and provides a compact medium for developing a schema. Also, CG can be used to represent the behavior of objects (although that is not a focus of this paper) and so can represent the object schemata produced by Object Oriented Analysis[8] and other methods. The advantage of CG is that it has a straightforward mapping into predicate calculus, so the formal meaning of each construction is unambiguous and clear. Finally, several researchers have adopted CG for various purposes in medical informatics.[29–32]

The schema is shown in two parts. The first part is the classification hierarchy of concept types (Fig. 3). Subtypes are shown indented under a supertype. The topmost types are **Person, Organization, Agent, Event, Physical Object**, and **Abstraction**. These are subtypes of the most general type "top," which is represented by the symbol **T** (not shown). Note that a given type can occur in more than one place in the hierarchy. For example, **Provider** is both a **Person** and an **Agent**. The second part of the schema consists of "canonical graphs," which define the conceptual relations that pertain to each concept (Fig. 4). Each relation connects a given concept to another concept in the schema. For example, **Service–Event** has a **recipient** (the patient on whose behalf events are performed), an **agent** (who performs the action), an **assistant** (a provider who assists the activity), a **recorder** (a provider who records information about the event), and a **location** (the place within the institution where the event occurs). Relations are "inherited" from their supertype. For example, **Event** has a start-time and an end-time. **Service** also has these relations, since **Event** is its supertype. This aspect of CG allows a very compact schemata to be created. Some concepts have been elaborated further in this schema. For example, **Number** and **Units** are not further defined.

Clinical events are the core of the conceptual schema;[11,33] they are events that involve the patient

```
[Patient : 12345] -
     (sex) -> [Gender : male]
     (birth-date) -> [Date : 1995-06-22].
```

Figure 2 Example of a conceptual graph in the linear format. The graph depicts a patient whose sex is male and whose birth date is 6/22/1995. Each concept appears in square brackets, and each relation appears in parentheses. Relations emanating from a concept are shown indented underneath, with an arrow pointing to the related concept.

```
Person
  Patient
  Provider

Organization
  Patient-Care-Organization
    Hospital
    Clinic
  Service-Department
    Laboratory
    Pharmacy

Agent
  Provider
  Organization

Event
  Service
    Laboratory-Procedure
      Laboratory-Panel
        Chem-7
        SMAC
      Laboratory-Test
        Serum-Sodium-Test
        Serum-Potassium-Test
    Drug-Administration
  Encounter
    Inpatient-Visit
    Outpatient-Visit
  Order
    Laboratory-Order
    Pharmacy-Order
Physical Object
  Chemical
    Potassium
    Sodium
  Drug
    Antibiotic
      Penicillin
      Ampicillin
  Specimen
    Serum
    Blood
  Measurable-Substance

Abstraction
  Temporal-Abstraction
    Frequency
    Date
    Date-Time
  Spatial-Abstraction
    Institution-Location
    Administration-Route
  Quantitative-Abstraction
    Quantity
    Number
    Units
```

Figure 3 Simple type hierarchy for clinical domain. Subtypes are indented under their supertypes. A subtype appears in multiple places in the outline when it has more than one supertype.

```
[Person]-
  (child-of) ->[Person]
  (spouse-of) ->[Person].
[Patient]-
  (birth-date) ->[Date]
  (sex) ->[Gender]
  (ssn) ->[SSN]
  (address) ->[Address]
  (telephone) ->[Telephone].
[Provider]-
  (employed-by) ->
  [Patient-Care-Organization].
[Event]-
  (recipient) ->[Patient]
  (agent) ->[Agent]
  (assistant) ->[Provider]
  (recorder) ->[Provider]
  (location) ->[Location]
  (start-time) ->[Date-Time]
  (end-time) ->[Date-Time].
[Service]-
  (agent) ->[Service-Department]
  (occurs-in) ->[Encounter]
  (ordered-as) ->[Order].
[Encounter]-
  (agent) ->
  [Patient-Care-Organization].
[Order]-
  (agent) ->[Physician]
  (quantity) ->[Number]
  (frequency) ->[Frequency]
  (event-start) ->[Date-Time]
  (event-end) ->[Date-Time]
  (occurs-in) ->[Encounter].

[Drug-Administration]-
  (agent) ->[Nurse]
  (quantity) ->[Quantity]
  (ordered-as) ->[Pharmacy-Order].
[Laboratory-Procedure]-
  (agent) ->[Laboratory].
  (ordered-as) ->[Laboratory-Order].
[Laboratory-Order]-
  (panel) ->[Laboratory-Panel].
[Pharmacy-Order]-
  (dose) ->[Quantity]
  (route) ->[Route]
  (drug) ->[Drug].

[Chem-7]-
  (contains) ->[ Sodium-Test]
  (contains) ->[Potassium-Test].
[Laboratory-Panel]-
  (contains) ->[Laboratory-Test].
[Laboratory-Test]-
  (measures) ->[Measurable-
[Substance]
  (samples) -> [Specimen]
  (value) ->[Quantity].
[Serum-Sodium-Test]-
  (measures) ->[Sodium]
  (samples) -> [Serum].
[Serum-Potassium-Test]-
  (measures) ->[Potassium]
  (samples) -> [Serum].
[Quantity]-
  (number) ->[Number]
  (units) ->[Units].
```

Figure 4 Simple canon of conceptual graphs for clinical domain. More complex graphs can be constructed from these canonical graphs.

during the provision of care. Clinical events include admission, discharge, clinic visit, drug administration, diagnostic procedures, therapeutic procedures, etc. Starting from the **Event** concept in the schema, note that certain key relations link clinical events with other concepts:

people: patient, health care providers, family members, insurance policy holders;

organizations: ancillary departments, inpatient administration, medical services, outpatient clinics;

physical objects: drugs, chemicals, specimens;

locations: patient rooms, emergency departments, operating rooms, nursing stations, physician offices;

abstractions: times, quantities.

Filtering Out Patient-Invariant Constructs

The primary purpose of the clinical repository (as defined in this paper) is to provide efficient access to the data of individual patients in both inpatient and outpatient settings. Therefore, the primary interest when implementing the repository is in the data which can vary from patient to patient. In developing the conceptual schema in the previous step, some constructs may be employed that provide background or explanatory information about other concepts in the schema but which do not vary from patient to patient. These "patient-invariant" constructs should be eliminated from the operational patient database to reduce redundancy.

For example, the conceptual schema in Figures 3 and 4 shows a number of concepts and relations that are useful in clarifying the meaning of other concepts but which are patient invariant. Laboratory tests have the relations **measures** and **samples**, these identify for each test the substance measured and the type of sample analyzed, respectively. These relations define exactly what the **Serum-Sodium-Test** concept is: a laboratory test that measures sodium levels in serum samples. In a given patient record, it is sufficient to record that the patient had a serum sodium test. Facts about what the test measures and the samples used are constant across all patients. Therefore, these relations need not be included in the repository schema used by the operational clinical information system.

This separation of highly detailed knowledge about clinical data from the essential facts about the care of patients results in a more generic schema. This step and the following one may give rise to a concern that semantic information is being lost; background knowledge about tests and drugs is clearly important and must be maintained somewhere. In many clinical information systems, this semantic information is maintained in the clinical data dictionary.[22,23] The dictionary will have a concept for each item of data that can be stored in the repository, such as laboratory tests, surgical procedures, and drugs. Any patient-invariant information that further defines the meaning of these concepts may also be stored there. Thus, any constraints that must be imposed on data as they are collected by applications and stored in the repository in the operational clinical information can be enforced using semantic information stored in the dictionary.

This division of the conceptual schema into two pieces—a generic repository and a dictionary of background information—has been chosen by many clinical information systems. It is the design feature that permits the most efficient retrieval of individual patient data and adaptation to changes in clinical applications.

Generic Transformations

After patient-invariant constructs have been eliminated, the conceptual schema can be greatly simplified by collapsing the many detailed concepts into a small set of generic concepts. In addition, the various detailed relations that connect concepts in the conceptual schema can be collapsed into generic "associations." This process of mapping highly detailed concepts into general ones is termed "generic transformation." In this paper, two important generic transformations are illustrated. The methods are shown through examples drawn from Figures 3 and 4, and the formal definitions of the transformations are given in Appendix A. These transformations are similar to those in Saltor et al.[34] but use the semantically rich model of conceptual graphs rather than the relational model.

The first generic transformation is called "flattening." This transformation is employed to reduce the classification hierarchy in the original schema, which has many levels, into a much smaller number of levels (often a single level). The second transformation is called "relation merging." This transformation combines several different relations in the conceptual schema into a single generic relation.

The result of applying generic transformations to the conceptual schema is a generic schema that is suitable for creating a database of patient information. Each transformation reduces the semantic content of the repository schema by making it more generic. In clinical information systems that have a sophisticated data dictionary, these semantic constraints are not lost but instead are transferred to the dictionary.

```
[Laboratory-Procedure-Event: P127]-
   (type)->[Chem-7]
   (agent) ->[Laboratory]
   (contains) ->[Laboratory-Procedure-Event: T005]
   (contains) ->[Laboratory-Procedure-Event: T006].

[Laboratory-Procedure-Event: T005]-
   (type)->[Serum-Sodium-Test]
   (agent) ->[Laboratory]
   (value) ->[Quantity].

[Laboratory-Procedure-Event: T006]-
   (type)->[Serum-Potassium-Test]
   (agent) ->[Laboratory]
   (value) ->[Quantity].
```

Figure 5 Three different instances of the Lab-Procedure-Event concept: a Chem-7 panel and two tests (Serum-Sodium-Test, Serum-Potassium-Test), which are part of the panel.

Flattening Example

Consider the hierarchy of laboratory procedure concepts in Figures 3 and 4. The types are **Lab-Procedure, Lab-Panel, Chem-7, SMAC, Lab-Test, Serum-Sodium-Test, and Serum-Potassium-Test**. In a real schema, there would be hundreds of panels and tests. When the flattening transformation (defined in Appendix A) is applied to this portion of the schema, the distinctions among these subtypes disappear. The many subtypes merge into a single new type, which can be called **Lab-Procedure-Event**. In addition, all the relations in these graphs are combined into one graph. As explained in the previous section, the relations for **measures** and **samples** are not included; they are considered to be descriptive information that belongs in the dictionary. All of these concepts collapse into the following single concept:

```
[Lab-Procedure-Event]-
      (type) →[Lab-Procedure]
      (agent) →[Laboratory]
      (contains) →[Lab-Procedure-Event]
      (value) →[Quantity].
```

This graph says that every **Lab-Procedure-Event** may contain other procedures (a property of panels) and that **Lab-Procedure-Event** may have a value (a property of **Lab-Test**).

Figure 5 shows three ways of filling in this graph, for a **Chem-7** panel, and two tests (**Serum-Sodium-Test, Serum-Potassium-Test**) that are part of the panel. This example demonstrates why it is necessary to add the new relation "**type**" to the **Lab-Procedure-Event** graph. Without this relation, it would be impossible to determine what kind of panel or test is denoted by each graph. In general, whenever a hierarchy of concepts is collapsed, it is necessary to add the new relation "**type**" in order to keep track of which specific member of the hierarchy is being referenced.

A complete conceptual schema would have hundreds of concepts for panels and individual laboratory tests. The generic schema need contain only one. Each time a **Lab-Procedure-Event** concept is stored in a patient record, the value of the relation "**type**" will be the identifier of one of the **Lab-Procedure** concepts. This identifier can be thought of as a pointer into the clinical data dictionary, where additional information about the panel or laboratory test may be obtained. The flattening transformation is the key to the flexibility of the repository: new identifiers can be added to the dictionary at any time, and the repository schema does not have to change.

Relation Merging Example

A given concept in the conceptual schema may have many detailed relations in it (directly or through inheritance). These relations become columns in a relational implementation (or fields in a file system, instance variables in an object-oriented system). Many detailed columns lead to inflexibility in the repository. As the clinical information system evolves, new columns must be added; this is extremely costly when tables are very large.[14,16] A solution to this design is to merge multiple relations into a single, generic relation.

As an example, consider the **Patient** entity in the conceptual schema in Figure 4. This concept has relations for the date of birth, sex, social security number, home address, and telephone number:

```
[Patient]-
      (birth-date) →[Date]
      (sex) →[Gender]
      (ssn) →[SSN]
      (address) →[Address]
      (telephone) →[Telephone].
```

This concept can be implemented in a relational database simply by creating a table, the columns of which have these names and data types. However, many other relations are possible for patient demographics, so administrators of the repository may find themselves frequently adding new columns to this table.

An alternative is to merge these relations into a single generic relation by applying the relation merging transformation (formally defined in Appendix A). The result is two concepts: the original **Patient** concept, which has a single generic relation, and a new concept called **Demographic**:

[Patient]-
(attribute) →[Demographic].
[Demographic]-
(type) →[Demographic-Type]
(value) →[Demographic-Value].

This transformation promotes the original relations (**birth date, sex, ssn, address, telephone**) into semantic types, which are made into subtypes of a new type called **Demographic-Type**. The various values that these relations could have are grouped under a new type called **Demographic-Value**. Figure 6 shows how this generic schema might be filled in. This schema is extremely simple and very flexible. Whenever a new demographic attribute is needed (e.g., ethnicity), it is necessary only to assign a new identifier for it in the dictionary. A relational implementation of this schema is shown in Table 2 and discussed in the following section.

For another example, consider the various events that can be related to a given **Service-Event** in the schema:

[Service-Event]-
...
(occurs-in) →[Encounter]
(ordered-as) →[Order-Event].

This graph states that a **Service-Event** may occur as part of an **Encounter** and may be requested by a given **Order**. A complete schema would contain many other relations to different types of events. To obtain a more generic schema, the relation merging transformation can be applied. This transformation merges the specific relations into a single generic relation. The result in this example is the statement that a **Service-Event** is related to one more **Related-Event** concept.

[Service-Event]-
...
(related-to) →[Related-Event].
[Related-Event]-
(type) →[Event-Relation]
(event) →[Event].

The relations "**occurs-in**" and "**ordered-as**" are promoted to semantic types, which are both subtypes of a new type, **Event-Relation**. When a particular **Related-Event** is filled in for a given patient's **Service-Event**, the value of the relation "**type**" is an identifier of one of the **Event-Relation** concepts (Fig. 7). This identifier serves as a pointer into the dictionary, where additional knowledge about event relations may be maintained. This knowledge might include a type hierarchy of event relations and information about exactly which events can be connected to one another and in what manner. The advantage of this schema is that new relations between events (e.g., causality) can easily be added by defining new instances of **Event-Relation** in the dictionary.

```
[Patient: 12345]-
    (attribute) ->[Demographic: D001]
    (attribute) ->[Demographic: D002]
    (attribute) ->[Demographic: D003].

[Demographic: D001]-
    (type) ->[Birth-Date]
    (value) ->[Date: 1995-06-22].

[Demographic: D002]-
    (type) ->[Sex]
    (value) ->[Gender: male].

[Demographic: D003]-
    (type) ->[SSN]
    (value) ->[Number: 123-45-6789].
```

Figure 6 Generic schema for patient demographics. The Patient concept can have any number of demographic attributes. Each demographic attribute has a type and a value.

Repository Implementation

The generic repository schema is easily converted into a database. The most practical choice now for a large-scale clinical information system in terms of speed and storage capacity is a relational database,[13-15] although some object-oriented databases may be considered. Each concept in the schema becomes a table in the relational database in the following way[7,10]:

1. The relations of a given concept that have atomic domains (Date, Date-Time, Number, etc.) become columns of the corresponding table, with the appropriate data type (date, timestamp, integer, etc.).

```
[Service: S001]-
    ...
    (related-to) ->[Related-Event: R001]
    (related-to) ->[Related-Event: R002].

[Related-Event: R001]-
    (type) ->[Occurs-In]
    (event) ->[Encounter: E001].

[Related-Event: R002]-
    (type) ->[Ordered-As]
    (event) ->[Order: O001].
```

Figure 7 Generic schema for relations between clinical events. The Service concept has two related events: an order and an encounter. The two Related-Event concepts specify the type of relation and the identifier of the related event.

2. The key of the table is either a column corresponding to the unique identifier of the concept or some combination of columns that determines a unique row of the table.

3. Conceptual relations that connect one concept to another in the schema become "foreign key" columns in the table. Additional columns must be added to correspond to the key columns of the referenced table.

As an example, consider the **Patient** concept as it is defined in the conceptual schema of Figure 4 (before the transformations of the previous section are applied). There are several atomic attributes: **birthdate, sex, ssn, address,** and **telephone**. These become columns with the following data types, respectively: date, character (fixed length 1), decimal (length 9), character (maximum length 256), and decimal (length 10). (In real systems, **Address** would probably be broken down into multiple columns for street, city, state, etc.) The unique identifier of **Patient** is the medical record number, so this becomes the primary key column of the table. In this database, all people (patients, providers, family members) can be grouped into a table called **Person**, in which common attributes (such as names) are stored. The supertype relation between **Patient** and **Person** is implemented as a foreign key by adding the unique identifier of the **Person** table (Person ID) as a column of the **Patient** table (Table 1).

Using the generic schema obtained by transformations in the previous section (Fig. 6), a generic table is created that can be called **Demographic** (Table 2). The conceptual relation "attribute" in the generic schema is implemented by adding a column for the patient identifier (a foreign key). In this way, a given patient may have any number of demographic attributes. (Further discussion of relation cardinality is beyond the scope of this paper.) The **Demographic-Type** column will contain identifiers for birthdate, sex, ssn, address, and telephone. The values for these various demographics are stored in the **Demographic-Value** column. Note that this column must store a wide variety of data types (date, character, decimal). There are several techniques to accommodate this diversity; the simplest is to define the column as character data of the maximum length necessary.

After generic transformations have been applied to a conceptual schema for patient data and then implemented as a relational database, the result is a database with a small number of tables. The implementation of a generic clinical repository at Columbia-Presbyterian Medical Center, operational now for five years, demonstrates that approximately two dozen generic tables are adequate to support a complex clinical information system.[11,20] This design is similar to that used by earlier systems, although these designs did not use relational technology.[1-5] Each generic table may eventually hold tens or hundreds of millions of rows of patient data if the repository is used as a longitudinal record from which no information can be deleted.

The database schema focuses on patient events; thus, most tables will have a column corresponding to the "**recipient**" relation in the repository schema. The domain of this column is patient identifiers (medical record numbers). Depending on the database software employed to implement the repository schema, the patient identifier column can be used to:

1. Index rows of the tables, providing fast access to patient events when a particular patient identifier is given.

2. Cluster rows on physical storage, further improving queries that retrieve data of a single patient by reducing the disk pages that must be examined by the database management system (DBMS).

3. Distribute data across different disks, enabling queries about different patients to execute in parallel.

4. Distribute data across different servers, reducing the transaction load on a given server.

One of the advantages of the generic design is that few relational joins are required when retrieving data. Queries that join data from multiple tables may involve a large number of disk accesses and may require several seconds or even minutes to complete. Because the generic design employs so few tables, related data tend to be stored in the same table.[17-19] For example, using generic transformations, all observational data, such as laboratories, radiology, pathology, drug administration, etc., may be stored in a single table. Any query that compares data in this table (e.g., laboratory and drug administration data) may be satisfied without an expensive join.

Discussion

How Generic Should the Repository Be?

Applying the transformations of Appendix A to the sample schema in Figures 3 and 4 yields the following repository schema:

[Concept]-
(related-to) →[Related-Concept].
[Related-Concept]-
(type) →[Relation]
(concept) →[Concept].

The dictionary would provide a type hierarchy of Concepts and Relations, with various names and identifiers for each, and it would contain semantics for the system, specifying which concepts are related to each other. This model could be implemented in a relational database with two tables. However, none of the techniques described in the previous section for physically managing these tables would apply, since there is no patient identifier. Thus, this schema is far too general.

At the other extreme, the schema in Figures 4 and 5 might be implemented directly as the clinical repository. This design might work well for a clinical research database, but it would perform poorly for patient care applications. Such a schema tends to spread a single patient's data across many tables. When data are retrieved, a large number of joins may be required, resulting in sluggish response times. A further problem is that this relational database would have very specific names for columns, derived from the conceptual relations in the schema (as is seen in the previous section for the **Patient** table). If any conceptual relation has been omitted from the schema, a table may have to be changed, and such schema changes can have an enormous impact.[14,16] With tables containing millions of rows, the cost of physically reorganizing data can be very high. If changes are frequent (typical in large, heterogeneous environments like academic medical centers), this architecture is unworkable.

Limitations of Generic Data Modeling

Generic data modeling entails an optimization of the clinical schema: the schema should contain as much semantics as possible to provide a useful structure for clinical events, but it should be sufficiently "open ended" to allow for flexibility. Experience with the clinical information system at Columbia–Presbyterian Medical Center suggests that a database schema based on clinical events having roughly the granularity discussed in this article (see Conceptual Schema for Clinical Data, above) results in a good tradeoff between flexibility and performance.[18]

However, these advantages still come at a certain price. The process of flattening the conceptual schema produces a useful generic schema, but constraints about the clinical domain are removed from the repository. In certain implementations, losing these constraints may introduce an inconsistent use of data in any computer applications that interact with the repository. For example, the generic schema may require that providers participate in clinical events, but it may not control the identifiers used for providers throughout the institution; without centralized control, different hospital departments may report which providers are involved in an event but may employ identifiers that differ from those of other departments. It was mentioned earlier in this article (see Filtering Out Patient-Invariant Constructs, above) that constraints that operate on a finer granularity than the clinical repository could, in principle, be enforced by a central data dictionary. For example, when storing data in the repository, such a centralized resource could be used to translate provider identifiers into a standardized representation (just as identifiers for medications and other data elements must be translated). Few clinical information systems have implemented this design to its fullest extent. Thus, large, heterogeneous organizations may have to tolerate a certain amount of inconsistency.

On the other hand, the lack of specificity that arises from generic data modeling has certain strategic advantages. Because the clinical domain is so complex, obtaining agreement across the institution on a wide range of data is a long and difficult process. The generic model can be viewed as the core that comprises those clinical data on which there is certain agreement and which are thought to change in structure at the slowest rate. This core enables integration efforts to get started, and it supports an incremental approach to large-scale information system development.[6]

This paper has drawn attention to the serious problems in creating clinical information systems that can adapt to changing requirements. Managing a patient database schema in a production environment over any significant period of time presents enormous design challenges. The behavioral aspects of clinical information systems ("processes," "methods," or "services") present even greater difficulties, since these tend to be the most volatile characteristic of an information system.[8] It is not clear that partitioning the conceptual schema into a repository of generic objects and a "dictionary" of specific objects can accommodate changes in system behavior. In the simple client-server architecture described in this paper, clients share repository data but are responsible for their own processing. In the future, architectures using "mediators" that combine both data and process in one schema may overcome these limitations.[16]

Conclusion

A clinical repository must support many diverse applications. The repository must provide continuous

access to data while it accommodates constant changes with respect to the needs of applications. In particular, the repository schema must be able to evolve without having an adverse effect on the applications. Traditionally, clinical information systems have addressed this problem by employing a generic schema for the repository, usually in combination with a dictionary of data elements.

In recent years, techniques of data modeling have evolved that enable the construction of large, detailed models of the information used in a domain. These formal techniques are essential in health care because of the size and complexity of this field. Although relational database technology has made tremendous improvements since the early days of clinical information systems, traditional approaches to database design have failed to produce information systems that can perform efficiently for large patient care applications. This paper offers a methodology that reconciles detail-oriented data modeling with performance-oriented database design. The techniques of generic data modeling may be employed to produce efficient, flexible database designs that can be used in operational systems.

References ■

1. Stead WW, Hammond WE. Computer-based medical records: the centerpiece of TMR. MD Comput. 1988;5:48–62.
2. McDonald CJ, Blevins L, Tierney WM, Martin DK. The Regenstrief medical record. MD Comput. 1988;5:34–47.
3. Whiting-O'Keefe QE, Whiting A, Henke J. The STOR clinical information system. MD Comput. 1988;5:8–21.
4. Grandia LD, Pryor TA, Willson DF, et al. Building a computer-based patient record system in an evolving integrated health system. In: Steen EB (ed). First Annual Nicholas E. Davies CPR Recognition Symposium. Washington, DC: Computer-based Patient Record Institute, 1995:3–32.
5. Curtis C. A computer-based patient record emerging from the public sector: the decentralized hospital computer program. In: Steen EB (ed). First Annual Nichols E. Davies CPR Recognition Symposium. Washington, DC: Computer-based Patient Record Institute, 1995:53–93.
6. Johnson SB, Forman B, Cimino JJ, Hripcsak G, et al. A technology perspective on the computer-based patient record. In: Steen EB (ed). First Annual Nicholas E. Davies CPR Recognition Symposium. Washington, DC: Computer-based Patient Record Institute, 1995:35–51.
7. Batini C, Ceri S, Navathe S. Conceptual Database Design: An Entity-Relationship Approach. Redwood City, CA: Benjamin/Cummings Publishing, 1992.
8. Coad P, Yourdan E. Object-Oriented Analysis, 2nd ed. Englewood-Cliffs, NJ: Prenctice-Hall, 1991.
9. Date C. Introduction to Database Systems, 6th ed. New York: Addison-Wesley, 1995.
10. Fleming C, Van Halle B. Handbook of Relational Database Design. New York: Addison-Wesley, 1989.
11. Johnson SB, Friedman C, Cimino JJ, Hripcsak G, Clayton PD. Conceptual data model for a central patient database. In: Clayton PD (ed). Fifteenth Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1992:381–5.
12. Gouveira-Oliveira A, Lopes L. Formal representation of a conceptual data model for the patient-based medical record. In: Safran C (ed). Seventeenth Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1994:466–70.
13. Simon AR. Strategic Database Management Technology: Management for the Year 2000. San Francisco: Morgan Kaufman, 1995.
14. Sjoberg D. Quantifying schema evolution. Information and Software Technology, 1993;13:35–44.
15. Dowgiallo E. A landmark year in review. Database Programming and Design, 1995, December (special supplement):54–6.
16. Wiederhold G. Modeling and system maintenance. In: Papazoglou N (ed). OOER '95: Object Oriented Relationship Modeling. Springer Lecture Notes in Computer Science. New York: Springer-Verlag, 1995(1021):1–20.
17. Friedman C, Hripcsak G, Johnson SB, Cimino JJ, Clayton PD. A generalized relational scheme for an integrated clinical database. In: Miller RA (ed). Fourteenth Symposium on Computer Applications in Medical Care. Washington, DC: IEEE Computer Society Press, 1990:335–9.
18. Johnson SB, Hripcsak G, Chen J, Clayton PD. Accessing the Columbia clinical repository. In: Ozbolt J (ed). Eighteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1994:281–5.
19. Dolin RH. A high-level object-oriented model for representing relationships in an electronic medical record. In: Ozbolt J (ed). Eighteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1994: 514–18.
20. Barrows RC, Johnson SB. A data model that captures clinical reasoning about patient problems. In: Gardner R. (ed). Nineteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1995:402–5.
21. Giordano R. Repeating datagroups: which way do we go? Database Programming and Design. 1992;5:47–51.
22. Cimino JJ, Hripcsak G, Johnson SB, Clayton PD. Designing an introspective, multi-purpose controlled medical vocabulary. In: Kingsland CL (ed). Thirteenth Symposium on Computer Applications in Medical Care. Washington, DC: IEEE Computer Society Press, 1989:513–18.
23. Linnarssen R, Wigertz OB. The data dictionary: controlled vocabulary for integrating clinical databases and medical knowledge bases. Methods Inf Med. 1989;28:78–85.
24. Van Ginneken AM, Stam H, Duiterhout JS. A powerful macro-model for the computer patient record. In: Ozbolt J (ed). Eighteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1994:496–500.
25. Essin DJ, Lincoln TL. Implementing a low-cost computer-based patient record: a controlled vocabulary reduces database design complexity. In: Gardner R (ed). Nineteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1995:431–5.
26. Sowa JF. Conceptual Structures: Information Processing in Mind and Machine. Reading, MA: Addison-Wesley, 1984.
27. Chen C. The entity relationship model: toward a unified view of data. ACM Transactions on Database Systems, 1977 (March).
28. Wiederhold G, El-Masri R. The structural model for database design. In: Chen C (ed). Entity Relationship Approach to System Analysis and Design. Amsterdam, The Netherlands: North Holland, 1980:237–57.
29. Friedman C, Cimino JJ, Johnson SB. A schema for repre-

senting medical language applied to clinical radiology. J Am Med Inform Assoc. 1994;1:233–48.
30. Campbell KE, Das AK, Musen MA. A logical foundation for representation of clinical data. J Am Med Inform Assoc. 1994;1:218–32.
31. Bell DS, Pattison-Gordon E, Greenes R. Experiments in concept modeling for radiographic image reports. J Am Med Inform Assoc. 1994;1:249–62.
32. Baud R, Lovis C, Alpay L, et al. Modeling for natural language understanding. In: Safran C (ed). Seventeenth Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1994:289–93.
33. Essin DJ, Lincoln TL. An information model for medical events. In: Ozbolt J (ed). Eighteenth Symposium on Computer Applications in Medical Care. Philadelphia: Hanley & Belfus, 1994:509–13.
34. Saltor F, Castellanos G, Garcia-Solaco M. Overcoming schematic discrepancies in interoperable databases. In: Hsoa DK, Neuhold EJ, Sacks-Davis R (eds). Interoperable Database Systems. New York: Elsevier, 1993:191–205.

APPENDIX A

Generic Transformations

Flattening

Given a canonical graph whose root is type A, together with the set of canonical graphs whose root types are B_1, B_2, . . . , B_n, such that B_i is a subtype of A, for $1 \leq i \leq n$:

[A]-
    (a_1) →[A_1]
    . . .
    (a_k) →[A_k].
[B_i]-
    (b_{i1}) →[C_{i1}]
    . . .
    (b_{im}) →[C_{im}].

replace these canonical graphs with a single canonical graph whose root is type A', such that the arcs of this graph are the union of the arcs of the original graph of type A, plus all the arcs of the graphs of type B_i:

[A']-
    (type) →[A]
    (a1) →[A1]
    . . .
    (an) →[An]
    (b_{11}) →[C_{11}]
    (b_{12}) →[C_{12}]
    . . .
    (b_{n1}) →[C_{n1}]
    (b_{n2}) →[C_{n2}]
    . . .
    (b_{nm}) →[C_{nm}].

An additional arc with a distinct name such as "type" must be added to the new graph. The value of this arc specifies which subtype of A is denoted by a particular instance.

Relation Merging

Given a canonical graph whose root is type A, with arcs a_1, a_2, . . . , a_n connecting to types (respectively) B_1, B_2 . . . , B_n:

[A]-
    (a_1) →[B_1]
    (a_2) →[B_2]
    . . .
    (a_n) →[B_n].

replace this canonical graph with the following graphs:

[A] →(r) →[R].

[R] →(a) →[B].
[A_1] →(a) →[B_1].
[A_2] →(a) →[B_2].
. . .
[A_n] →(a) →[B_n].

Where R is a generic relation for A, and B is the least supertype of B_1, B_2, . . . , B_n. Each original relation a_i is promoted to a type A_i, where each A_i is a subtype of R. These graphs maintain the original restrictions on what can be connected to what.

This type hierarchy of relations can then be further compressed using the flattening transform above:

[A] →(r) →[R'].

[R']-
    (type) →[R]
    (a) →[B].